

AD-A155 488

USE AND DESIGN OF AN ACTIVE DATA DICTIONARY FOR LOCAL
VALIDATION OF INPUT DATA(U) NAVAL POSTGRADUATE SCHOOL
MONTEREY CA R M DIBONA MAR 85

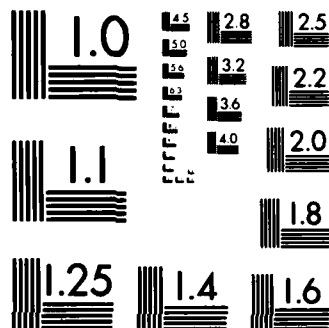
1/1

UNCLASSIFIED

F/G 9/2

NL

					END							
					FILED							
					DATA							



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(2)

AD-A155 488

NAVAL POSTGRADUATE SCHOOL

Monterey, California



DTIC
ELECTE
JUN 27 1985
S B

THESIS

USE AND DESIGN OF AN ACTIVE DATA DICTIONARY
FOR LOCAL VALIDATION OF INPUT DATA

by

Robert M. DiBona

March 1985

Thesis Advisor:

D.R. Dolk

Approved for public release; distribution is unlimited

DTIC FILE COPY

85 6 7 125

DISCLAIMER NOTICE

**THIS DOCUMENT IS BEST QUALITY
PRACTICABLE. THE COPY FURNISHED
TO DTIC CONTAINED A SIGNIFICANT
NUMBER OF PAGES WHICH DO NOT
REPRODUCE LEGIBLY.**

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A155488	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Use and Design of an Active Data Dictionary For Local Validation of Input Data		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis March 1985
7. AUTHOR(s) Robert M. DiBona		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE March 1985
		13. NUMBER OF PAGES 57
		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution is unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Data Dictionary, Data Integrity, Data Dictionary		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This thesis addresses the problem of small user groups being forced to use input data collected and processed by sources outside their span of control. Specifically, the use of an active data dictionary to locally validate such input is examined. The thesis proceeds from a general review of data validation techniques and criteria, through an examination of data dictionaries, to an illustration of how an active data (Continued)		

ABSTRACT (Continued)

dictionary can be configured to act as a "data filter" for input data.

Key initial planning and design steps are set forth, including requirements analysis, data definition, and initial logical design. A checklist of questions to answer during each of these activities is included.

The concepts discussed in the paper are then applied to a specific case (DCSPLANS Branch, U.S. Army Military Personnel Center, Alexandria, VA) resulting in a "data filter" structure diagram that is tailored to the DCSPLANS' environment and their unique validation needs.

Accession For		
DTIC TAB	<input checked="" type="checkbox"/>	
HTOL	<input type="checkbox"/>	
Unannounced	<input type="checkbox"/>	
Justification		
Distribution/		
Availability Codes		
Avail and/or		
Dist	Special	
A1		

Approved for public release; distribution is unlimited.

Use and Design of an Active Data Dictionary
For Local Validation of Input Data

by

Robert M. DiBona
Major, United States Army
B.S., Boston College, 1972
M.Ed. Boston University, 1976

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

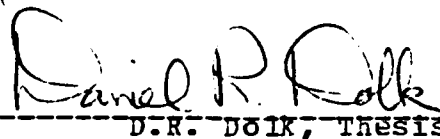
from the

NAVAL POSTGRADUATE SCHOOL
March 1985


Author:

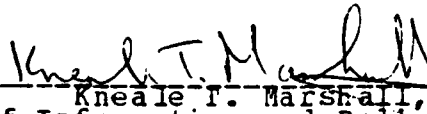

Robert M. DiBona

Approved by:


D.R. Dolk, Thesis Advisor


Norman R. Lyons, Second Reader


Willis R. Greer, Jr., Chairman,
Department of Administrative Sciences


Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

This thesis addresses the problem of small user groups being forced to use input data collected and processed by sources outside their span of control. Specifically, the use of an active data dictionary to locally validate such input data is examined. The thesis proceeds from a general review of data validation techniques and criteria, through an examination of data dictionaries, to an illustration of how an active data dictionary can be configured to act as a "data filter" for input data.

Key initial planning and design steps are set forth, including requirements analysis, data definition, and initial logical design. A checklist of questions to answer during each of these activities is included.

The concepts discussed in the paper are then applied to a specific case (DCSPLANS Branch, U.S. Army Military Personnel Center, Alexandria, VA) resulting in a "data filter" structure diagram that is tailored to the DCSPLANS' environment and their unique validation needs.

TABLE OF CONTENTS

I.	INTRODUCTION	8
	A. CONTROL OF DATA	8
	B. DCSPLANS, MILPERCEV	9
	C. THESIS METHODOLOGY	11
II.	INPUT VALIDATION	13
	A. GENERAL DESCRIPTION	13
	B. VALIDATION TECHNIQUES	14
	1. Category	14
	2. Transaction Validation	15
	3. Format Checks	15
	4. Reasonableness Checks	16
	C. EDIT AND VALIDATION RULES	18
III.	DATA DICTIONARY AS "DATA FILTER"	19
	A. BASIC CONCEPTS	19
	1. Data Dictionary	19
	2. Metadata	19
	3. "Active" Data Dictionary	20
	4. Data Extraction	22
	B. CONFIGURATION	23
	1. Metadata Generation	23
	2. Edit and Validation Programs	24
	3. General Design	25
	C. ADVANTAGES	27
IV.	PLANNING AND GENERAL DESIGN	29
	A. KEY DEVELOPMENT PHASES	29
	B. PHASE ONE - SYSTEM ENVIRONMENT/GENERAL CHARACTERISTICS	30

1. Description	30
2. Checklist	30
C. PHASE TWO - DATA DEFINITION/VALIDATION	
CRITERIA	32
1. Description	32
2. Checklist	32
D. PHASE THREE - INITIAL LOGICAL DESIGN	33
1. Description	34
2. Checklist	34
3. Follow-on Design	36
V. THE DCSPLANS "DATA FILTER" SYSTEM	37
A. DCSPLANS SYSTEM GOAL AND OBJECTIVES	37
B. "DATA FILTER" STRUCTURE	40
1. Structure Diagram	40
2. Narrative Descriptions	43
C. "DATA FILTER" IMPLEMENTATION	50
VI. CONCLUSIONS AND RECOMMENDATIONS	52
A. CONCLUSIONS	52
B. RECOMMENDATIONS	54
LIST OF REFERENCES	56
INITIAL DISTRIBUTION LIST	57

LIST OF FIGURES

1.1	Force Plans Branch	10
3.1	Passive Dictionary	21
3.2	Active Dictionary	22
3.3	Data Extraction Design	24
3.4	General Data Filter Design	25
3.5	The Data Filter System	26
4.1	Structure Diagram	35
5.1	Sample Paragraph Format	41

I. INTRODUCTION

A. CONTROL OF DATA

One problem plaguing today's information manager is the serious lack of control over data which has developed as computers and their applications have spread throughout organizations. Recently, there has been a considerable increase in the attention being paid to this problem. However, most organizations whose information systems were developed in the 60's and early to middle 70's still suffer the ill effects of improperly controlled data. In these environments, redundant, incomplete, and inaccurate data are still prevalent. Under such circumstances, the probability that faulty data will directly contribute to poor organizational planning and ineffective decision-making is significantly increased.

While some organizations have undertaken action to correct their data control problems, many others are overwhelmed by the enormity, complexity, and cost of the task. In very large organizations, the cost and complexity take on proportions that appear extremely prohibitive. Unfortunately, it is these large organizations which have the greatest need for carefully controlled data. Large organizations are also more likely to experience adverse effects which extend beyond those found in smaller enterprises.

One of these effects is manifest in the helpless position in which some organizational user groups find themselves. As one cog in a large wheel, these groups often

are forced to use data collected and processed by other organizational elements over whom they exercise no control. A serious danger in this circumstance is the receipt and subsequent use of inaccurate data.

Information systems need valid data to be effective! A rash assumption by a data processing element that inaccurate data are correct can have devastating effects on a parent organization, especially if information based on the data is used for strategic planning/decision-making.

When input data of unknown validity is being transferred among data processing elements within an organization, the problem is almost always a systemic one with deep and widespread roots. Corrective action on an organization-wide basis often is neglected because of excessive costs. Users who find themselves in these situations are frequently left to their own devices, and they must develop their own means for validating inputs. An illustration of a user group experiencing such a situation is the Office of the Deputy Chief of Staff, Plans (DCSPLANS), U.S. Army Military Personnel Center (MILPERCEN), in Alexandria, Virginia.

B. DCSPLANS, MILPERCEN

U.S. Army MILPERCEN is responsible for the worldwide distribution and professional development of Army officer and enlisted personnel. Within MILPERCEN, DCSPLANS has the mission of planning, programming, and executing current and future force alignment, i.e., matching personnel inventory to force authorization levels.

DCSPLANS is composed of five branches, each of which monitors a specified portion of the force alignment mission.

FORCE PLANS BRANCH
DAPC-CPF

INPUTS TO MODELS
=====

ENLISTED MASTER FILE
OFFICER MASTER FILE
GAIN/LOSS TAPE
PERSONNEL MANAGEMENT AUTHOR-
IZATION DOCUMENT
SPACE IMBALANCE MILITARY
OCCUPATIONAL SPECIALTY FILE
RETENTION RATES REPORT
SELECTIVE REENLISTMENT
BONUS LEVELS
TRAINEES, TRANSIENTS, HOLDERS,
STUDENTS REPORT



MODELS
=====

PERSONNEL READINESS INDICATOR MODEL
PERSONNEL POLICY PROJECTION MODEL
OFFICER PROMOTION MODEL
MANNING INDICATOR MODEL
OBJECTIVE FORCE MODEL
OFFICER ASSET UTILIZATION MODEL
OFFICER STRENGTH MANAGEMENT MODEL
CORRECTABLE AUTHORIZATIONS DATA BASE
MODEL

Figure 1.1 Force Plans Branch

Each branch uses a series of computerized models to perform a variety of forecasting functions. See Figure 1.1 for an example of the models and input files used by DCSPLANS' branches. Many of these models are quite complex and draw

input data from both MILPERCEN and non-MILPERCEN sources. Some input files are extremely large, feed a number of models, and historically, have been prone to error. None of the input files are under DCSPLANS control.

The output of DCSPLANS' models is used for crucial top level decision making which will determine the structure and content of army forces well into the future. As such, the DCSPLANS output must exhibit a very high degree of validity. Currently, however, DCSPLANS is unable to verify the accuracy of much of the input data being used by its models. Thus, despite the correctness of the models themselves, the reliability of the DCSPLANS product must be considered doubtful.

DCSPLANS officials are quite concerned about their present inability to insure that the data used in their models are accurate. They realize the problem will not be solved for them soon by the organization (MILPERCEN), and that they must devise their own local solution. A variety of options are available to them. Some are quite poor (e.g., maintain the status quo and rely on the input data sources to insure validity); others are more feasible, but still contain serious shortcomings (e.g., update/convert every DCSPLANS model to include its own validation process). A much more effective and efficient alternative is described in this paper, i.e., the use of an active data dictionary as a "filter" to validate input before the data is processed by the various models.

C. THESIS METHODOLOGY

This thesis will explore the concept of using an active data dictionary as a local validation tool. It will proceed

from a general review of data validation, through an examination of data dictionaries and their design, to an illustration of how an active data dictionary can be beneficially applied to DCSPLANS operations.

Chapter Two of the thesis cites the essential role of data validation as an integral part of a data processing system. Validation criteria and techniques used in the "data filter" are reviewed, and the general nature of edit and validation rules is introduced.

Chapter Three explores the data dictionary. It includes some basic definitions and concepts, and specifically addresses how an active data dictionary is used to validate data.

Chapter Four outlines an approach to "local" initial design of a data dictionary "filter" system. This chapter also includes a recommended "checklist" of questions a user group can ask to define its own data dictionary/validation requirements and system structure.

Chapter Five specifically addresses the DCSPLANS situation. It cites a proposed goal and some key objectives of a DCSPLANS validation system, and uses a modified structure diagram of a "data filter" to illustrate the recommended approach to DCSPLANS' data validation dilemma.

Chapter Six summarizes the results of this thesis.

II. INPUT VALIDATION

A. GENERAL DESCRIPTION

Inaccurate data items can easily find their way into master files and databases, either through direct input by users or through improper processing actions by application programs. Regardless of origin, inaccurate data are poison in any ADP system. Information created from inaccurate data also tends to be inaccurate, and decisions based upon such information are counterproductive to organizational goals in almost every instance. Data is a valuable resource, and its accuracy is crucial to organizational success.

Validation is that set of actions which attempts to preclude the existence of inaccurate data within an information system. Validation tests can be implemented at any number of stages within the data processing cycle: prior to input, upon input, during processing, and after processing (output checks). "Input validation", as implemented by an active data dictionary system, occurs at the second stage.

Input validation focuses specifically on data being entered into a system. Its aim is to detect errors and thereby insure the initial accuracy of the master file or database being constructed/updated. [Ref. 1:p. 326] During input validation, checks are conducted to insure that the input/update operation itself is legal, and that input data does not violate prescribed accuracy constraints. Creation of a new file or the update of an existing one is a processing stage that demands extremely careful data validation, especially in those cases where the input data

is received from sources outside the control of the processing element. Fortunately, it is at this stage that the accuracy of data can be checked most effectively [Ref. 2:p. 289]. One additional caution which must be mentioned at this point is that data does not become inaccurate from entry errors alone. Data may be inaccurate simply because it is old! Previously accurate values may no longer be correct because available new values have not superseded older values due to neglected updates. Validation processes also must check for these types of inaccuracies.

B. VALIDATION TECHNIQUES

1. Category

The general category of input validation techniques used by the "data filter" being proposed examines input data in the exact form in which it arrives for processing. The techniques involved detect errors by checking the "acceptability" of both the data transactions and the data itself. This checking is accomplished through a series of programmed instructions/rules, and is implemented very effectively by an active data dictionary system. Three basic techniques are included in the category: transaction validation, format checks, and reasonableness checks. A well designed validation program includes a combination of all three. [Ref. 3:p. 248]

The transaction validation technique is used to verify the legitimacy of transactions which input data. The format checks and reasonableness checks, on the other hand, are used to examine the correctness of data items themselves. In order to facilitate a clearer picture of the "data filter" design which will be presented in the next two

chapters, a brief description of the three validation methods is provided below.

2. Transaction Validation

Transaction validation should be the first technique to be applied. It certifies that " a specific transaction is one that can be processed by the system and is being submitted properly." [Ref. 4:p. 218] Its focus is the verification that the type and purpose of the transaction are legitimate processing actions, and that the originator of the transaction has the authority to initiate it. Transactions determined to be inaccurate are rejected.

Related validation checks which also must be conducted during this juncture of the processing cycle are checks for sequential dependencies and/or proper timing. For example, a Monthly_Report transaction may not be able to take place until Monthly_Update transactions are successfully executed.

The role of transaction validation as a "first step" stems from the potential damage which could be inflicted upon a system by the processing of an invalid transaction. Even if the invalid transaction is subsequently discovered, recovery may prove extremely difficult. An ounce of prevention, in this case, is certainly worth a pound of cure!

Once transaction validity is established, the input data itself is examined through a series of format checks and reasonableness checks.

3. Format Checks

Format checks compare the actual contents of a field to a pre-set series of user-defined rules. A record, whose

contents fail to conform to the prescribed format, either is rejected outright or transferred to an appropriate error handling routine. Some of the more common format checks are:

- a) Length Checks: used to verify that a field contains a prescribed minimum, maximum, or fixed amount of characters.
- b) Character Type Checks: used to verify that a field contains only specifically authorized value types, i.e., numerics only, alphabets only, blanks, or special characters.
- c) Character Pattern Checks: used to verify that the contents of a field match a prescribed pattern of alphabets, numerics, dashes, etc.
- d) Date Checks: used to insure that the contents of a date field are entered in the required, standard format, i.e., YYMMDD or YYDDD.

4. Reasonableness Checks

Reasonableness checks test data items to insure that data values fall within the limits of established constraints. These constraints are separated into three basic types. Field constraints limit the value of a given data item. Intrarecord constraints limit values between fields in the same record. Interrecord constraints limit values between fields in different records. [Ref. 5: p. 179] Reasonableness checks based upon field constraints are fairly straightforward in design and application. Intrarecord and interrecord constraint checks, however, deal with logical accuracy and the interrelationships among data items. As such, they are much more difficult to develop and manage. Common reasonableness checks are:

- a) Field Constraints

- Range Checks - used to verify that the field value falls within a specified range, i.e., the value does not violate an upper or lower limit.
- Sequence Checks - used to test a specially created field to insure records are processed in the proper order. These checks are also used to verify the presence of all required records.
- Completeness Checks - used to confirm that each mandatory field in a record is filled with a data item of some prescribed size.
- Date Checks - used to verify that the contents of a date field do not violate earliest or latest acceptable date restrictions.
- Code Checks - used to verify that the contents of a code field are contained within a listing of valid and current codes.

b) Intrarecord and Interrecord Constraints:

- Completeness Checks - used to identify those fields in a record which must be filled based upon the contents of other fields in that record (intrarecord) or other records (interrecord).
- Consistency Checks - used to verify that the values in certain fields are valid in relation to the data values of other fields (either in the same record or other records).

An example of an intrarecord completeness check is, "If the Conversion Indicator field in a record is filled, then the Conversion Code field in that record must also be filled." An interrecord version of a completeness check is as follows: "If the VRE Multiplier field is filled for any record in this run, then all VRE Multiplier fields must be filled."

An example of an intrarecord consistency check is, "If the POS Code in a record is 63H, then the grade value in

the record must be either E4 or E5." An interrecord consistency check is "no SSN field value may be the same as the SSN field value of another record."

It is also possible to have "interfile" dependencies, e.g., a record with an SSN field value of "99999999" in file "A" must have the same MOS field value as a record in file "B" which has an identical SSN field value of "99999999."

C. EDIT AND VALIDATION RULES

There must be an organized and consistent method for applying the validation checks cited above to data being input into an information system. The vehicle for this application is the edit and validation rule (EVR). EVRs are explicit statements of constraints about the data in a system. These rules monitor the basic structure and relationships of data items, and enforce processing restrictions established by the information manager.

[Ref. 6: p. 146]

Two key issues concerning EVRs must be addressed when building a data validation system. The first is how to properly develop consistent rules. Consistent rules promote accurate data, whereas contradictory rules produce an unreliable data system that eventually will crash.

(Definition and development of EVRs will be covered in chapter four as an integral part of the overall "data filter" design process).

The second key issue is where to place an EVR module, (i.e., is it better to embed it as part of an application program, or is it better to make it a separate validation program?). The use of an active data dictionary as a "data filter" argues for the latter approach. The rationale for SUCH A PLACEMENT IS SET FORTH IN THE NEXT CHAPTER.

III. DATA DICTIONARY AS "DATA FILTER"

A. BASIC CONCEPTS

Four basic concepts are central to a clear understanding of how a data dictionary can be used locally to validate data maintained and provided by other sources. These are: Data Dictionary, Metadata, "Active" Data Dictionary, and Data Extraction.

1. Data Dictionary

A data dictionary is a centralized repository of all definitive information about the relevant data in an enterprise. The data dictionary provides the user a description of what data exists, what it looks like, and what it means. [Ref. 7:p. 1] A data dictionary can be as simple as a manual catalog system or as complex as an automated set of programs which controls a wide range of the enterprise's data processing operations.

2. Metadata

The real world of an enterprise contains a number of data objects (entities) which are represented in the enterprise's information system as data elements, records and files. For example, customers (entity) are represented by a set of data elements/fields (CUST_ID, CUST_NAME, etc.) which comprise records (CUST_REC), which, in turn, are grouped into files (CUST_FILE). The data used to define and describe these entities are called metadata, i.e., data

about the data. Metadata are stored in the data dictionary, forming a metadata database or metadatabase. [Ref. 8:p. 9] Dictionary metadata contain the characteristics of each data object. The metadata answer the following questions:

- a) What data is available in the enterprise?
- b) What does the data mean?
- c) How is the data structured?

d) What constraints and relationships exist? Typically, dictionary metadata include: object name, short name, synonym or aliases, source, narrative description, records/files that use or contain the data object, data structure/format, integrity constraints (e.g., value range), and relationships/dependencies. [Ref. 9:p. 18] Metadata are essential ingredients in the validation of data by a data dictionary system.

3. "Active" Data Dictionary

There are two basic modes in which a data dictionary can function: passive or active. A passive data dictionary merely registers the metadata and provides the user a facility for interactive query and/or report generation. It does not require that data processing operations depend upon it for metadata, and no direct link is maintained between the passive data dictionary and other system components. (See Figure 3.1) In fact, application programs and processes may obtain their metadata entirely from other sources.

An active data dictionary, on the other hand, exercises a great deal of control over processing and metadata usage within an information system. A data dictionary is said to be active with respect to an information system, if, and only if, that system is dependent upon the data dictionary for its metadata. (See Figure 3.2)

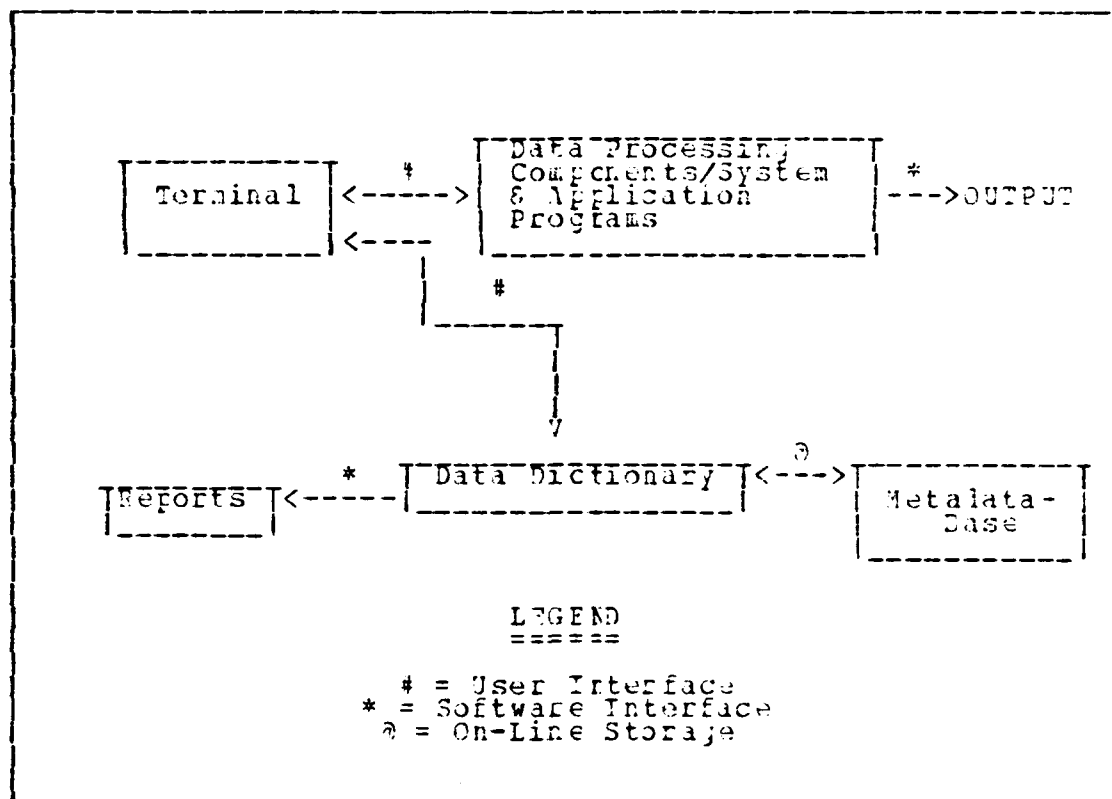


Figure 3.1 Passive Dictionary

A dictionary is active to a lesser degree when only some of the system's programs and processes are dependent upon it for metadata. The more programs or processes that rely on the dictionary, the more active it is said to be. [Ref. 10:p. 22] The value of an active data dictionary stems from the establishment of mandatory interfaces between it and various system processes. When the data dictionary is used as a "data filter", these mandatory interfaces will insure that input data conform to pre-defined rules and standards.

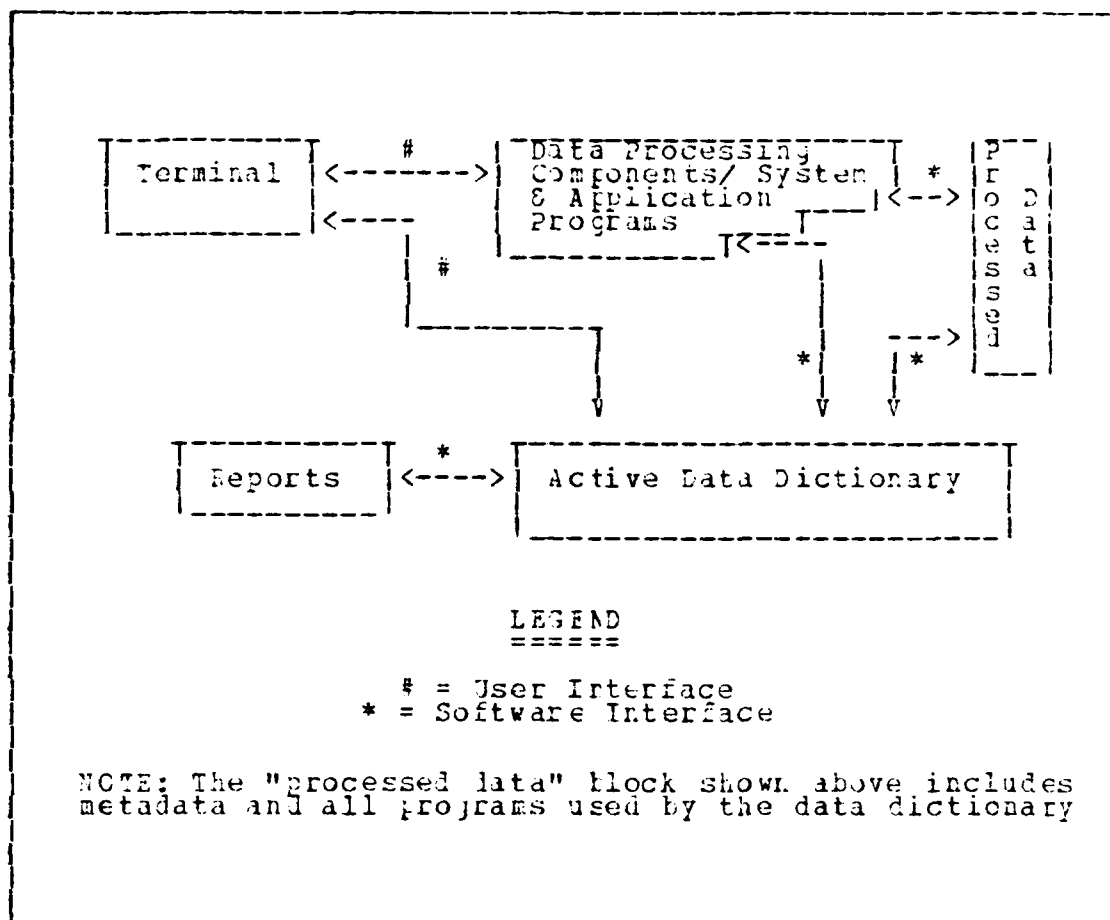


Figure 3.2 Active Dictionary

4. Data Extraction

Data extraction is a technique whereby a subset of data from a very large file system or database is transferred to a much smaller "extracted" file or database. The data extraction process can be either quite simple or very complex. A complex data extraction process is designed to collect, format, and integrate data from a number of source files/databases into a single data source whose contents are specifically tailored to the needs of a single user or group of users. Such a system involves extensive data

description, subsetting, aggregation, and presentation operations. [Ref. 11:p. 245] This thesis addresses data extraction from a much simpler perspective, i.e., as a means to limit the size of the data to be validated by the data dictionary. In most cases, user applications do not need all data contained in a large data source. Thus, the extraction of only pertinent data (a much smaller subset), usually serves to increase the speed of application programs acting upon the data. Such data extraction operations can be used to greatly enhance the efficiency of the proposed "data filter" when large source files are involved. A diagram of a simple data extraction design which can be used in conjunction with a data dictionary "filter" is shown in Figure 3.3.

Throughout the remainder of this thesis, the term "data filter" will refer to the active data dictionary validation system being proposed.

B. CONFIGURATION

1. Metadata Generation

The key to constructing the data filter is incorporating into a data dictionary the capability to generate the metadata needed by a system's edit and validation software. The metadata generation is triggered by the edit and validation software through the issuance of commands and applicable parameters. The data filter must be designed so that the edit and validation, with its mandatory call for metadata generation, is automatically activated during all data input operations. The resulting metadata generation produces data descriptions based upon the characteristics stored in the data dictionary metadatabase. These data descriptions are transformed into specific edit

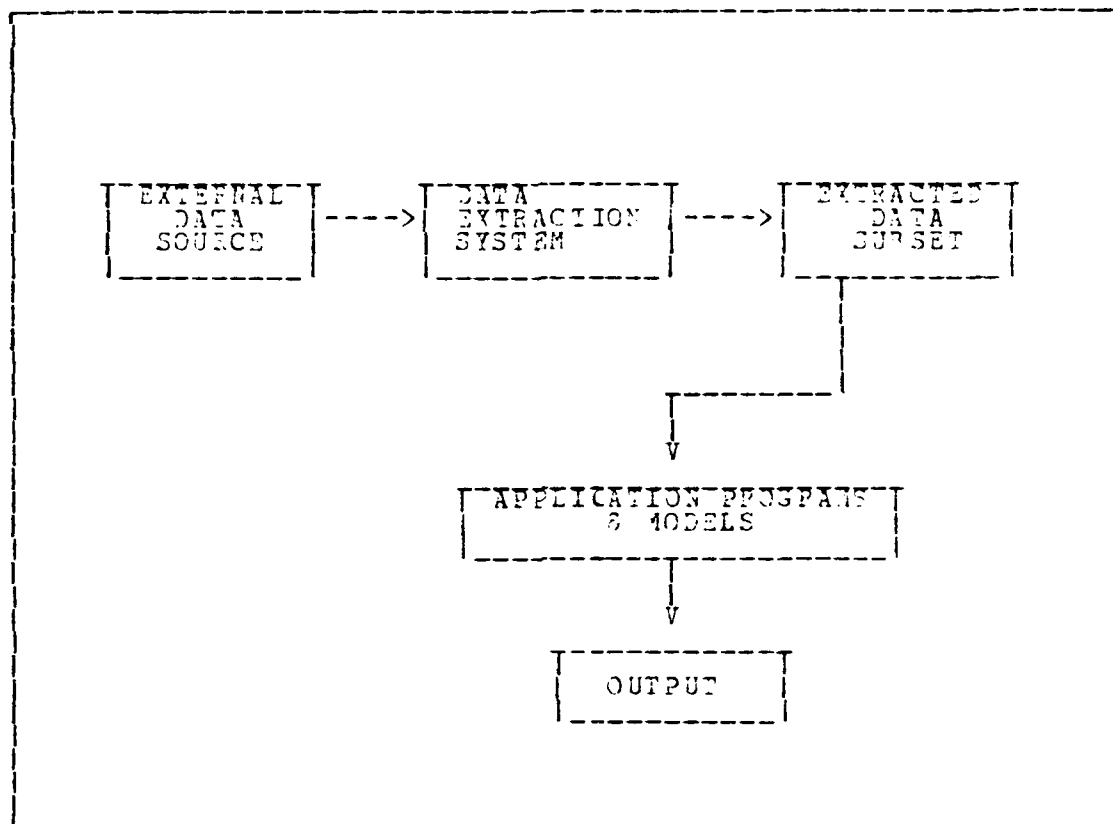


Figure 3.3 Data Extraction Design

and validation rules (EVR) for use by the edit and validation programs. [Ref. 12:p. 116]

2. Edit and Validation Programs

Edit and validation programs are separate from the application programs which enter the data into the system. They cannot be executed without data dictionary metadata (in the form of EVR) through which they will filter all incoming data. These programs are usually general purpose in nature. The tailoring of the programs to specific types of data is accomplished through the EVR provided by the active data dictionary. For example, an EMP data entry operation will

result in different EVR being passed to an edit and validation program than will a PMAD data entry (PMAD data may be composed of totally dissimilar data objects than PMF data, and may also involve very different validation criteria). Various edit and validation programs can be incorporated into the data filter to accommodate distinct categories of data entry operations, e.g., updates, deletions, creation of new files, etc.

3. General Design

Figure 3.4 depicts a generalized data filter design. The data dictionary generates metadata based upon commands

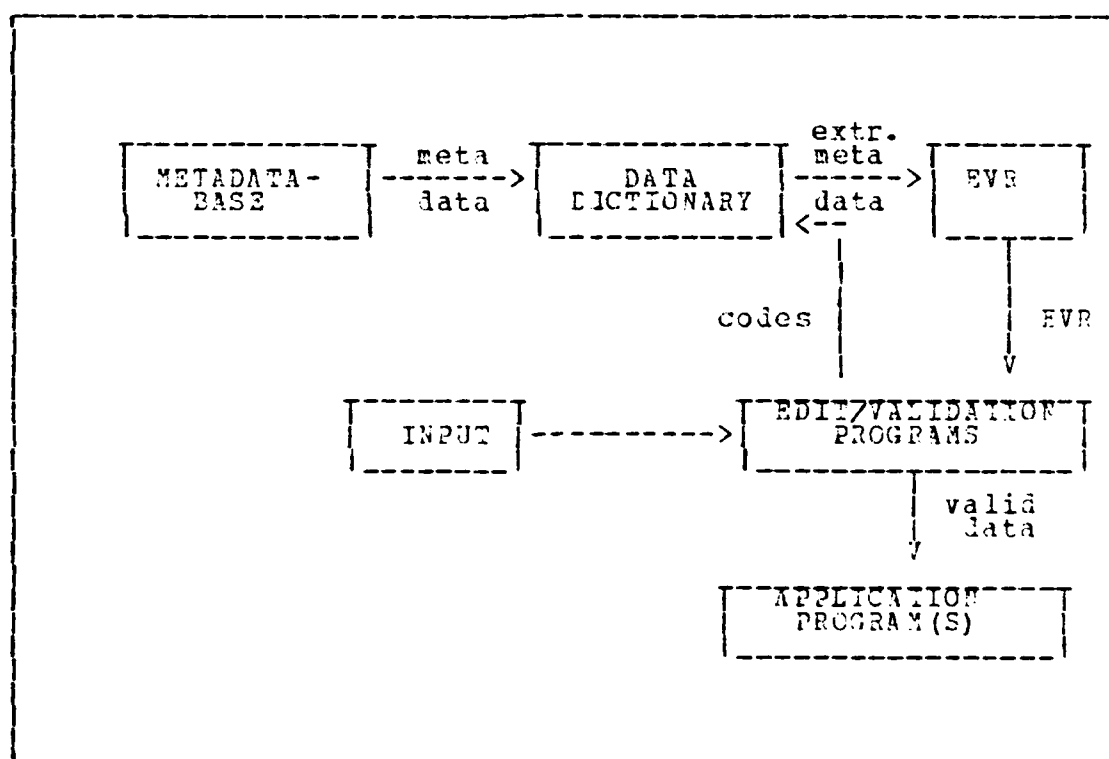


Figure 3.4 General Data Filter Design

from the edit and validation program. Then, the metadata is transformed into EVR which are fed back into the edit and validation program. The edit and validation program "filters" incoming data through the EVR during the edit and validation process. "Correct" data is moved to the appropriate storage area, and erroneous data is either rejected outright or sent to an error file for future editing and resubmission.

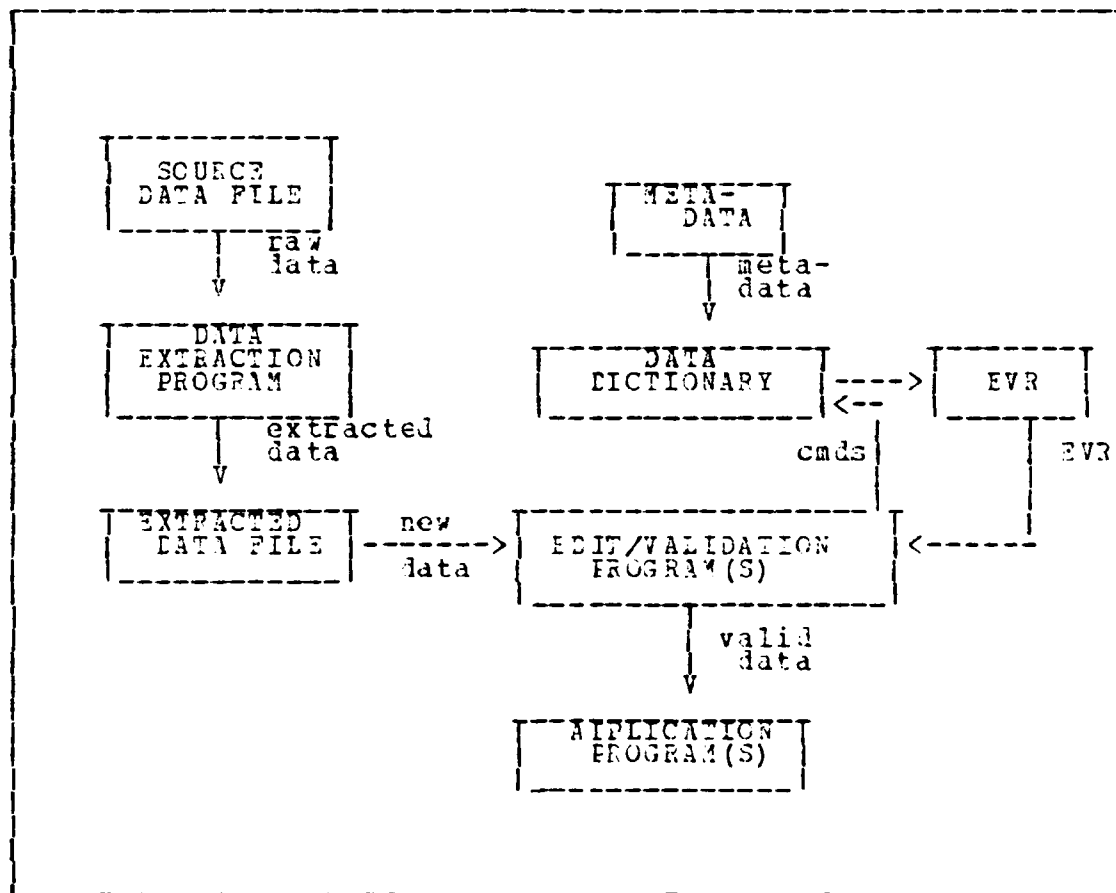


Figure 3.5 The Data Filter System

Figure 3.5 shows the complete data filter system with a data extraction module added. This configuration

increases data validation efficiency by reducing the amount of data to be "filtered." In the DCSPLANS' case, due to the enormity of the ENF and some other source data files, the time saved becomes quite significant.

C. ADVANTAGES

Almost all data editing and validation systems provide the user a capability to validate and edit data, and to correct and report erroneous data. There are, however, added benefits to be gained by using the active data dictionary approach which forms the basis of the data filter configuration described above.

First, since the active data dictionary becomes the sole source of metadata for all edit and validation processes, redundant metadata is eliminated and metadata consistency is promoted. In essence, a much greater degree of control over metadata is realized, and, as a result, regulated, consistent validation of data is achieved.

Second, the data dictionary affords the user a very flexible and easily adjustable validation mechanism. Changes in data and revisions to validation criteria do not require modification of application programs or edit and validation programs. Instead, changes are easily accommodated by simple adjustments to metadata/ENF.

Third, should the information system involved be file-based (as is the case with DCSPLANS), the data dictionary approach is an invaluable "bridge" for a future transition to a database system. Ease of transition is promoted by already having in existence an organized, centralized store of the enterprise's metadata.

One other benefit of the proposed data filter system stems from the separation of the data extraction program

from the actual edit and validation activities. Not only is overall validation speed increased, but also the user now has the option, in exigent circumstances, to forego validation entirely if time constraints demand such action. An interdependent extraction/validation process would not allow this alternative.

IV. PLANNING AND GENERAL DESIGN

A. KEY DEVELOPMENT PHASES

A software product's ability to do what it is supposed to do efficiently is largely governed by the quality of the detailed design and coding that creates it. In turn, successful detailed design and coding are directly tied to the quality of initial planning and design activities. Thus, the planning and preliminary design steps taken by users to develop a local data filter are crucial, and must be comprehensively and carefully accomplished.

Planning and initial design of a data filter is a three phased process. Phase one describes the system's environment and general characteristics. Phase two develops data definitions and validation criteria. Phase three produces an initial logical design of the system. A description of each of these phases is presented below, along with a "checklist" of relevant questions which serves as a guide for proceeding through the phase.

The checklists form a framework within which users/developers can methodically develop the data filter. The framework assists them in:

1. Obtaining a clear, comprehensive picture of the environment in which the data filter will function.
2. Identifying and defining the data to be validated, and determining the nature and scope of validation required.
3. Constructing well-defined, functionally structured validation and EVR modules.

B. PHASE ONE - SYSTEM ENVIRONMENT/GENERAL CHARACTERISTICS

1. Description

This phase identifies all hardware and firmware being used (or projected for use) in the overall information system, and describes its environment (e.g., distributed vs. centralized system, file system vs. database system, etc.). It notes validation capabilities already built into the system, and also identifies commercial validation capabilities which are compatible with existing hardware and firmware.

Phase one also uncovers the general nature of the input data to be validated. It identifies the broad categories of input data, examines data stability and consistency, and looks at who exercises control over the entry of data into the system. This phase outlines data entry methods and notes the various processing stages at which data validation may occur (pre-input, during input, etc.). An overview of system output is also formulated. The level of accuracy required for the output is established, and the degree to which output validity is dependent upon valid input is determined.

2. Checklist

Answers to the following questions will provide a clear picture of the overall system, including inputs and outputs:

- a) What major hardware components comprise the system?
- b) What operating system is used?
- c) What validation capabilities are already built into the system hardware/firmware?
- d) Are there currently any plans to change/expand major system hardware?

- e) Are any system-compatible data validation products currently available (either in-house or commercially)?
- f) What system-compatible data dictionary software is currently available (either in-house or commercially)?
- g) Are we dealing with a file-based or database system?
- h) What portions of the information system are distributed?
- i) How stable are system inputs? (i.e., Are different data elements, records and files added or deleted on a frequent basis?)
- j) Are data definitions and parameters changed frequently?
- k) Are we dealing with a stable number of data elements which will retain stable attributes?
- l) Is input processed in a batch mode, on-line, or both?
- m) Is any pre-input validation conducted? Describe!
- n) Is any output validation conducted? Describe!
- o) What are the sources of input data? Identify all input files and the applications for which they provide data.
- p) What degree of control over the entry and update of input data is exercised by system users?
- q) From what locations, and by whom, can data be added, changed or deleted.
- r) What sources beyond the user's control provide input data? Identify the data provided by each of these outside sources.
- s) How often is data entered? Updated?
- t) How is the processed data being used? (A general description, e.g., report generation, modeling, etc.)
- u) For each application, report, etc., how critical is validity? (i.e., What are the consequences of inaccurate outputs?)

C. PHASE TWO - DATA DEFINITION/VALIDATION CRITERIA

1. Description

This phase identifies and defines the system's data entities. For the purpose of the data filter, data entities include all data elements entered into the system and the records and files which contain them. The applications which use/process these entities are also established.

Phase two also sets forth all validation checks required. Data element characteristics such as description, range, type, size, sequence, etc. are recorded, and all entity relationships are carefully delineated. The information developed during this phase forms the data dictionary metadatabase, and is used to construct the system's EVF and validation program modules.

2. Checklist

Answers to the questions listed below will enable the user/developer to identify, describe, and determine the interrelationships of all system entities. He will also be able to establish validation criteria for each entity and cross-reference them to the applications which require that such validation occur.

- a) What data elements does the system contain?
- b) What record(s) contain these data elements?
- c) What file(s) contain these records?
- d) For each application (model):
 - Which files feed it data? Which records?
 - Which data elements does it use/process?
 - Which data elements must be validated (i.e., does the validity of the application's output depend on this input data element being valid)?
 - Is a specific sequence of data entry required?

- What pre-entry updates/transactions must occur, if any?
- e) For each data element:
 - What is its name? Any Synonyms or aliases?
 - What is its Short Name/Programming Name?
 - What is its ID#?
 - What is its character type (alpha, numeric, etc.)?
 - What minimum and maximum number of characters are allowed?
 - What numeric value range applies?
 - What character pattern is used (e.g., CCC-NNN-CC)?
 - Is there a minimum/maximum range of allowable change from one update to the next?
 - What cause and effect relationships exist with other data elements? In the same record/file, in other records/files? (e.g., If "A" is changed, then "B" must be changed).
 - Is a particular update sequence required?
 - Do date fields have any earliest or latest date limits?
 - Do date fields require a special format (e.g. YYMMDD)?
 - What direct relationships exist with other data items? (e.g., value of "A" must always be twice that of "B").
 - Is the data element a code or a value that be checked against a table or listing of valid codes or values?

D. PHASE THREE - INITIAL LOGICAL DESIGN

1. Description

Phase three produces a model of the logical structure of the data filter system which later will be "built" (during coding and testing). Since it forms the basis for all further design steps and refinements, this preliminary logical design is the key step in the data filter design process. The data filter structure developed during this phase is based upon the general filter design cited in chapter three and the system environment and data/validation information gathered during phases one and two.

Phase three gives the user a description of the data filter system goal and objectives, and presents the major system functions. These major functions are then decomposed into sub-functions until a series of single, independent modules have been identified. This overall system architecture is depicted in a hierarchical structure diagram (See Figure 4.1) accompanied by narrative descriptions of the modules.

2. Checklist

Answers to the following questions will enable the user/developer to produce the information described above:

- a) What is the goal of the system? (State the general long-term desired effect).
- b) What are the system's key objectives? (Enumerate the critical milestones to be accomplished to satisfy the stated system goal).
- c) What are the system's major functions? (List the general processing activities required to meet system objectives). For example, a bank's checking account system may have four major system functions: (1)

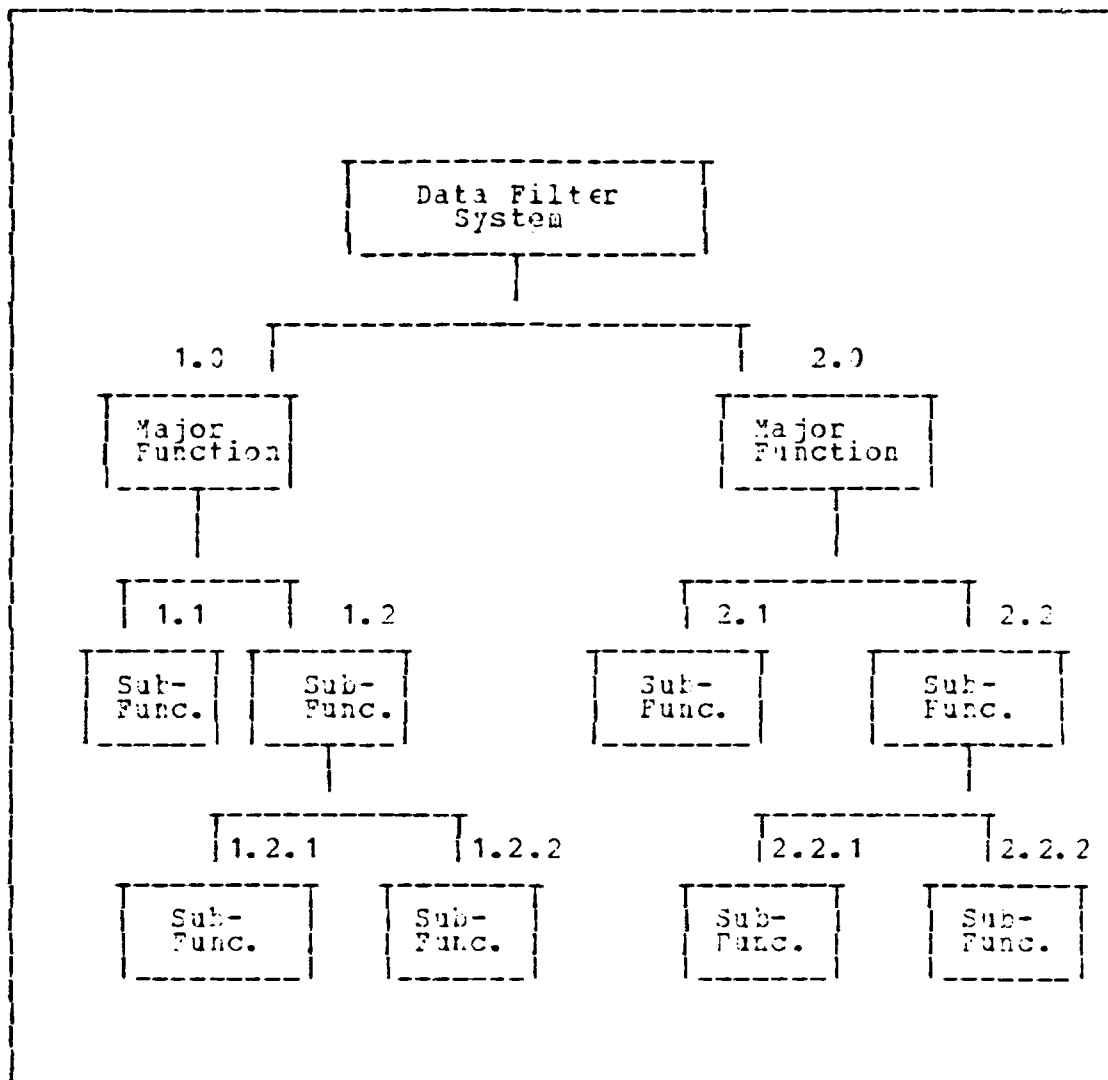


Figure 4.1 Structure Diagram

performing account administration (open accts., close accts, etc.) (2) processing deposits, (3) processing withdrawals, (4) maintaining an account transaction database.

- 1) What modules (sub-functions) comprise each of the system's major functions? (Limit to no more than 3-5 modules per function, and repeat the process level by

level until no further module decomposition is necessary, i.e., simple, independent modules have been created).

- e) What does each system module do? (Give a precise, concise description of approximately two sentences).

3. Follow-on Design

Once the above phases have been completed and carefully documented, the data filter structure has been tailored to the user's specific environment and validation needs. Subsequent development involving detailed design (data flows, data stores, interfaces, etc.), coding, testing, etc. can follow using one of a number of applicable methodologies which currently exist.

V. THE DCSPLANS "DATA FILTER" SYSTEM

This chapter specifically addresses the DCSPLANS' "data filter" system. It provides a statement of the system's overall goal and its key objectives. It also expands the general data filter design provided in chapter three into a more detailed hierarchical design structure tailored to the DCSPLANS situation.

A. DCSPLANS SYSTEM GOAL AND OBJECTIVES

A number of DCSPLANS' unique operational characteristics must be considered when formulating the system's goal and its key objectives. These critical aspects are uncovered during Phases I and II of the preliminary development activity (presented in the previous chapter), and are used to create the Phase III deliverables illustrated in this chapter (System Goal/Objectives and Structure Diagram with Narratives). A sample of the DCSPLANS characteristics having the greatest impact on the general system design are presented below.

The most important fact is that DCSPLANS personnel have little faith in the accuracy of input data they are receiving from a variety of very large source files prepared and maintained by elements outside their span of control. At the present time, DCSPLANS does not possess the capability to validate this questionable input data. They are, however, extremely worried about the adverse impact of such input data on the validity of model outputs.

Input source files provide crucial data to DCSPLANS' force alignment models. Each of the files needs a varying

number of models, and supplies a unique set of data elements depending on the particular model involved. Generally, the data elements contained in the source files and the data elements required by the models remain the same, creating relatively good system stability in this regard. There are, however, occasional changes made in the data elements provided or required. A DCSPLANS validation tool must provide the flexibility to incorporate such changes easily.

In many cases, models using the same data elements from the same source file require different degrees of validation (e.g., the validity of input data element "A" from the Enlisted Master File may be crucial to the validity of Personnel Readiness Indicator Model output, but inconsequential to the validity of output produced by the Personnel Policy Projection Model (PPM)). Thus, a DCSPLANS validation tool must be able to differentiate between the validation required for Enlisted Master File data when used by the Personnel Readiness Indicator Model as opposed to the PPM, and it must apply edit and validation rules accordingly.

Generally, DCSPLANS' models are run on a standard schedule which coincides with required briefings/reports and which also facilitates use of one model's output as input for another model. There are, however, occasions when a model's output is required on very short notice. In these circumstances, the time normally devoted to data validation may not be available, and the DCSPLANS' models would have to be run in the quickest possible time without regard to data integrity. While such a procedure seems unwise, it may occur, and the DCSPLANS validation tool must provide for such a contingency by allowing itself to be circumvented if required. In this regard, the DCSPLANS data filter cannot be a mandatory part of any integral data extraction or modeling process.

The majority of DCSPLANS modeling activities will be done in a batch mode. The extraction of pertinent data from large input files is also a batch process (e.g., the "UTSACS" program developed and used by DCSPLANS to extract pertinent data from the Enlisted Master File). However, capabilities to manipulate data dictionary metadata on-line and to query the metadatabase on-line are crucial to effective, user-friendly operation of the data filter system. All other data filter processes (e.g., EVE formulation) will be done in batch mode to insure run-time efficiency.

Based upon an examination of the overall DCSPLANS situation, and keying on the points just mentioned, the goal of the DCSPLANS data filter system is to validate all externally provided input data used by DCSPLANS' force alignment models in consonance with established DCSPLANS quality control standards.

Key objectives of the DCSPLANS data filter system are:

1. It must be compatible with the existing DCSPLANS computer system configuration.
2. It must allow flexible and easy additions and updates to the metadatabase.
3. Its interface with the data extraction and modeling processes must be optional (at the discretion of the Chief, DCSPLANS; otherwise it will be an automatic, mandatory interface).
4. It must provide for the automatic adjustment of edit and validation rules to suit the particular source file and model being processed.
5. It must provide an interactive on-line query facility for accessing the metadatabase.
6. It must provide an error/status report generation facility.
7. It must be a user-friendly system.

8. System development and implementation costs must be consistent with the "local" nature of the system. A conservative approach is desired.

B. "DATA FILTER" STRUCTURE

This section uses a structure diagram (in modified format) to set forth the proposed structure of the DCSPLANS "data filter" system software. The structure is derived from a functional decomposition process in which major system functions are split successively into sets of sub-functions. The proposed DCSPLANS system will be decomposed to three levels. This decomposition demonstrates the hierarchical control structure and relationships of modules which comprise the overall "data filter" program. It does not represent any particular processing sequence or order of decision-making. [Ref. 13:p. 149]

The structure diagram is normally presented in the graphical format shown in Figure 4.1. However, due to the crowding effect that will occur from a three-level decomposition, the major system functions (level 1) and subordinate modules (levels 2 and 3) are represented here in paragraph/sub-paragraph format (See Figure 5.1). Modules depicted in this manner are easily transferred to a graphic representation of the overall system, if required.

1. Structure Diagram

The proposed data filter system contains five major functions (Control Data Filter System, Maintain Metadatabase, Produce EVs, Validate Input Data, Generate Reports). The system's hierarchical structure is illustrated below, followed by descriptions of each major function, sub-function, and lower level module.

DCSPLANS Data Filter

- 1.0 FIRST MAJOR FUNCTION (Level 1)
 - 1.1 First Sub-function of 1.0 (Level 2)
 - 1.1.1 First Module of 1.1 (Level 3)
 - 1.1.2 Second Module of 1.1 (Level 3)
 - 1.1.3 Third Module of 1.1 (Level 3)
 - 1.2 Second Sub-function of 1.0 (Level 2)
 - 1.2.1 First Module of 1.2 (Level 3)
 - 2.0 SECOND MAJOR FUNCTION (Level 1)
 - 2.1 First Sub-function of 2.0 (Level 2)
 - 2.1.1 First Module of 2.1 (Level 3)
 - 2.1.2 Second Module of 2.1 (Level 3)
 - 2.2 Second Sub-function of 2.0 (Level 2)
 - 2.3 Third Sub-function of 2.0 (Level 2)
 - 2.3.1 First Module of 2.3 (level 3)
- (ETC.)

Figure 5.1 Sample Paragraph Format

- 1.0 CONTROL DATA FILTER SYSTEM
 - 1.1 Verify Transaction Validity
 - 1.1.1 Read Access and Transaction Codes
 - 1.1.2 Evaluate Codes
 - 1.1.3 Implement Validity Decision
 - 1.2 Provide Menu/Screen
 - 1.2.1 Read Validity Decision
 - 1.2.2 Display Appropriate Screen
 - 1.3 Transfer Control
 - 1.3.1 Read Screen Input
 - 1.3.2 Determine Proper Process
 - 1.3.1 Pass Program Control
- 2.0 MAINTAIN METADATABASE
 - 2.1 Control

- 2.1.1 Provide Metadatabase Menu
 - 2.1.2 Transfer Control
 - 2.2 Add Metadata
 - 2.2.1 Read Add Data
 - 2.2.2 Check Uniqueness
 - 2.2.3 Check Format
 - 2.2.4 Accept Data
 - 2.3 Delete Metadata
 - 2.3.1 Read Delete Request
 - 2.3.2 Locate Metadata
 - 2.3.3 Remove Metadata
 - 2.4 Change Metadata
 - 2.4.1 Read Change Request
 - 2.4.2 Locate Metadata
 - 2.4.3 Update Metadata
- 3.0 PRODUCE EVR
 - 3.1 Control
 - 3.2 Retrieve Metadata
 - 3.2.1 Read Source File/Model Codes
 - 3.2.2 Open Metadata File(s)
 - 3.2.3 Extract Pertinent Data Values
 - 3.3 Formulate EVF
 - 3.3.1 Load Variables
 - 3.3.2 Set Switches
- 4.0 VALIDATE INPUT DATA
 - 4.1 Control
 - 4.2 Select EVF
 - 4.2.1 Determine Input Record Types
 - 4.2.2 Extract Applicable EVF
 - 4.3 Apply EVR
 - 4.3.1 Read Input Data
 - 4.3.2 Read EVF
 - 4.3.3 Check Parameters
 - 4.4 Provide Processed Input Data

- 4.4.1 Read Error Code
- 4.4.2 Transfer Erroneous Data/Error Code
- 4.4.3 Transfer Valid Data
- 4.5 Maintain Statistics
 - 4.5.1 Maintain Transaction Count
 - 4.5.2 Maintain Error Count
 - 4.5.3 Sort Error Types
- 5.0 GENERATE REPORTS
 - 5.1 Control
 - 5.2 Retrieve Report/Response Data
 - 5.2.1 Determine Report/Response Type
 - 5.2.2 Read Applicable Data
 - 5.3 Perform Calculations
 - 5.4 Provide Report/Response
 - 5.4.1 Determine Format
 - 5.4.2 Format Data
 - 5.4.3 Transfer to Output Device

2. Narrative Descriptions

The following are succinct explanations of the key aspects of each structure diagram function, sub-function, and module. Each lower level description serves to refine/expand the detail of its superior level.

- 1.0 CONTROL DATA FILTER SYSTEM: This function controls access to the data filter system and verifies transaction validity. It also provides screens for implementing other major system functions, and transfers control to these processes.
- 1.1 VERIFY TRANSACTION VALIDITY: This sub-function insures that the user is authorized access to the system for the desired transaction, and that the transaction itself is valid (e.g., an attempt to validate the Enlisted Management File for use in the Officer Promotion Model would be rejected).

- 1.1.1 READ ACCESS AND TRANSACTION CODES: This module reads in the user's access code and the transaction codes indicating the desired process and the source input file/model(s) involved.
- 1.1.2 EVALUATE CODES: This module checks user-supplied codes against authorized access and transaction codes.
- 1.1.3 IMPLEMENT VALIDITY DECISION: This module will either reject the transaction or pass an indication of a valid transaction to module 1.2.1. This module also sets restrictions within authorized processes (e.g., a user may be allowed to add metadata, but not change or delete existing metadata).
- 1.2 PROVIDE MENU/SCREEN: This sub-function provides the user with the appropriate screen for continued use of the system.
 - 1.2.1 READ VALIDITY DECISION: This module reads the validity indicator produced by module 1.1.3.
 - 1.2.2 DISPLAY APPROPRIATE SCREEN: This module causes either a menu or screen, as appropriate, to appear on the monitor.
- 1.3 TRANSFER CONTROL: This sub-function passes control to an appropriate system module in response to user input.
 - 1.3.1 READ SCREEN INPUT: This module reads user responses to terminal prompts.
 - 1.3.2 DETERMINE PROPER PROCESS: This module interprets user input in terms of the desired system function (e.g., update metadata, generate report, etc.).
 - 1.3.3 PASS PROGRAM CONTROL: This module passes control to the appropriate system module.
- 2.0 MAINTAIN METADATABASE: This function creates new metadatabase entries, deletes metadatabase contents, and makes changes to the existing metadatabase.

- 2.1 CONTROL: This sub-function displays the metadatabase menu, and governs the activation and sequence of add, change and delete processes.
- 2.1.1 PROVIDE METADATABASE MENU: This module displays a menu giving the user options of adding, deleting or changing metadata.
- 2.1.2 TRANSFER CONTROL: This module passes control to either modules 2.2, 2.3, or 2.4, depending on user's request and access authorization.
- 2.2 ADD METADATA: This sub-function reads metadata input, checks it for duplication and proper entry format, and either rejects the input or stores it in the metadatabase.
- 2.2.1 READ ADD DATA: This module reads data which the user desires to enter into the metadatabase.
- 2.2.2 CHECK UNIQUENESS: This module checks metadatabase to insure data to be added does not already reside there.
- 2.2.3 CHECK FORMAT: This module checks data to be added for compliance with prescribed standard metadata entry formats.
- 2.2.4 ACCEPT DATA: This module evaluates results of module 2.2.2 and 2.2.3 processing, and either rejects data to be added or stores it in the metadatabase.
- 2.3 DELETE METADATA: This sub-function reads metadata deletion request, locates the data in the metadatabase, and removes it.
- 2.3.1 READ DELETE REQUEST: This module reads the user's request to delete data.
- 2.3.2 LOCATE METADATA: This module locates indicated metadata in the metadatabase.
- 2.3.3 REMOVE METADATA: This module removes metadata from the metadatabase after a re-verification of the user's desire to delete the data.

- 2.4 CHANGE METADATA: This sub-function reads a metadata change request, locates the data to be changed, and updates the data after verification that the new metadata meets the prescribed entry format.
- 2.4.1 READ CHANGE REQUEST: This module reads the user's request to update existing metadata.
- 2.4.2 LOCATE METADATA: This module locates the metadata to be changed.
- 2.4.3 UPDATE METADATA: This module replaces old metadata with new metadata.
- 3.0 PRODUCE EVR: This function produces edit and validation rules for use by sub-function 4.3. Metadata values are extracted from the metadatabase and are transformed into bounded conditional statements through which input data will be run.
- 3.1 CONTROL: This sub-function governs the activation and sequence of processes involved with the production of edit and validation rules.
- 3.2 ACCEPT PROCESSING CODES: This sub-function reads the source file and model codes entered by the user, opens appropriate metadata files, extracts applicable metadata values, and stores them in a "variables" file.
- 3.2.1 READ SOURCE FILE/MODEL CODES: This module reads the source file and model identification codes entered earlier by the user.
- 3.2.2 OPEN METADATA FILE(S): This module identifies and opens all metadata files containing data relating to source file and models noted by module 3.2.1.
- 3.2.3 EXTRACT PERTINENT DATA VALUES: This module extracts pertinent metadata values from opened metadatabase files and stores the data in a "variables" file.
- 3.3 FORMULATE EVR: This sub-function reads the metadata values stored in the "variables" file into a

file of pre-established conditional statements, thereby setting switches either on or off and setting upper and lower boundaries of acceptable input data values.

(Setting and boundaries will therefore vary according to the combination of source file and model codes presented by the user.)

- 3.3.1 LOAD VARIABLES: This module reads the "variables" file into a file of pre-set conditional statements.
- 3.3.2 SET SWITCHES: This module, depending on variable values, sets switches either on or off and establishes upper and lower boundaries, as required.
- 4.0 VALIDATE INPUT DATA: This function actually performs the validation by selecting specific EVR, applying these EVR to the input data, and providing the processed input data to either a "validated data" file or an "error" file. This function also maintains statistics on the number of data items processed and the number and category of errors found.
- 4.1 CONTROL: This sub-function governs the activation and sequence of processes involved in the actual validation of input data.
- 4.2 SELECT EVR: This sub-function identifies the type of record(s) being validated from the source file, and activates only those EVR which apply. (This sub-function precludes the validation program from unnecessarily running an input record past all source file EVR, thereby enhancing run-time efficiency of the overall process.)
- 4.2.1 DETERMINE INPUT RECORD TYPES: This module identifies the subset of records that are being validated from the source input file.
- 4.2.2 EXTRACT APPLICABLE EVR: This module extracts only those EVR applicable to the record types being validated.

- 4.3 APPLY EVR: This sub-function reads the input data and its associated EVR, and compares them to verify compliance.
- 4.3.1 READ INPUT DATA: This module sequentially reads input data to be validated.
- 4.3.2 READ EVR: This module reads EVP from module 4.2.2.
- 4.3.3 CHECK PARAMETERS: This module compares input data to EVR parameters, assigning an appropriate error code (including "no error").
- 4.4 PROVIDE PROCESSED INPUT DATA: This sub-function reads the processed data and its error code, and transfers the data accordingly.
- 4.4.1 READ ERROR CODE: This module reads the data and associated error code from module 4.3.3.
- 4.4.2 TRANSFER ERRONEOUS DATA/ERROR CODE: This module transfers erroneous data with its associated error code to an "error" file.
- 4.4.3 TRANSFER VALID DATA: This module transfers all valid input data to a "validated data" file.
- 4.5 MAINTAIN STATISTICS: This sub-function maintains a running count of the number of transactions processed and the number and type of errors found.
- 4.5.1 MAINTAIN TRANSACTION COUNT: This module maintains a running count of the number of transactions processed in a validation activity.
- 4.5.2 MAINTAIN ERROR COUNT: This module counts the number of errors found and notes the error code involved.
- 4.5.3 SORT ERROR TYPES: This module sorts a validation activity's error count by type of error.
- 5.0 GENERATE REPORTS: This function accepts requests for both printed reports and interactive (terminal) responses, determines and retrieves the appropriate

report/response data, performs calculations and formatting as required, and issues the requested report/response.

- 5.1 CONTROL: This sub-function governs the activation and sequence of processes involved with the production of printed reports and interactive response to terminal queries.
- 5.2 RETRIEVE REPORT/RESPONSE DATA: This sub-function determines the type of report/response desired and reads required data from appropriate files.
 - 5.2.1 DETERMINE REPORT/RESPONSE TYPE: This module interprets the user request for information in terms of report/response content.
 - 5.2.2 READ APPLICABLE DATA: This module locates, reads and temporarily stores the data needed for the requested report/response.
- 5.3 PERFORM CALCULATIONS: This sub-function determines whether calculations are required to produce desired information, and if so, it reads the appropriate data and performs the required operations, producing "new" report/reponse data.
- 5.4 PROVIDE REPORT: This sub-function determines the appropriate report/response format, formats the data accordingly, and transfer the formatted data to the appropriate output device.
 - 5.4.1 DETERMINE REPORT FORMAT: This module determines the format required for the desired response in accordance with pre-established format parameters.
 - 5.4.2 FORMAT DATA: This module arranges data in proper format.
 - 5.4.3 TRANSFER TO OUTPUT DEVICE: This module transfers the formatted data to the appropriate output device.

C. "DATA FILTER" IMPLEMENTATION

Two key advantages inherent in the proposed local data validation system concept are low development costs and speedy implementation. In this light, initial DCSPLANS development efforts must focus on the creation of a prototype system that takes maximum advantage of existing resources. Specifically, the DCSPLANS prototype must incorporate the existing UTRACS program which extracts relevant Enlisted Master File (EMF) data, the existing DBASE II data dictionary which currently includes general model and office metadata in its metadatabase, and the existing DCSPLANS IBM PC microcomputer. The DCSPLANS local data filter system therefore will consist of an IBM PC based, DBASE II program which filters EMF input data for use in two application models (two models must be used to test the system's ability to differentiate between the degrees of validation required by separate models using the same input data source file).

The following steps suggest a methodology for development of the initial DCSPLANS prototype "data filter" system.

1. Determine and implement the proper interface mechanism for feeding UTRACS extracted EMF data through the IBM PC data filter system.
2. Expand current data dictionary capabilities by creating additional metadatabase modules which will accept and store metadata about source file and model data elements. Create an additional data dictionary metadatabase module that will accept and store EVR.
3. Using the Phase II checklist from chapter four, comprehensively construct data definitions for EMF and model data elements, and create the EVR metadata which sets data element validation parameters and

interrelationships. This step must be accomplished with the full, constant cooperation of those DCSPLANS personnel most closely acquainted with the EMF and the two application models being used for the prototype.

4. Load the data definition and DVS metadata into the data dictionary metadata base.
5. Using the functional modules from section 3 of this chapter as a guide (particularly function 4.0), create an edit/validation program which will control and implement the overall data filter process.

The development methodology presented above is based upon a limited on-site review of DCSPLANS operations. A more comprehensive examination of the DCSPLANS environment (See Phase I of the planning and initial design process described in chapter four) will most likely uncover some additional requirements and necessary adjustments. Therefore a detailed on-site environmental review is an essential prerequisite to any DCSPLANS data filter development/implementation effort, especially one being undertaken by non-DCSPLANS personnel.

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

DCSPLANS, MILPERCEN suffers from a data control problem common to many small user groups in large data processing systems. It is unable to verify the correctness of input data obtained from sources outside its span of control. At the present time, DCSPLANS must rely almost exclusively on the competence of its outside sources to guarantee the integrity of its input data. The situation is causing DCSPLANS' managers a great deal of concern.

Top-level Army decision-makers use output from DCSPLANS' applications to formulate long-range personnel management policies. Thus, the adverse impact of erroneous input data entering DCSPLANS' models can be far-reaching and extremely serious. Despite this fact, DCSPLANS' small size relative to the overall MILPERCEN information processing system precludes it from strongly influencing the adoption of a system-wide validation capability. DCSPLANS must therefore develop and implement a "local" solution to its data validation problem.

DCSPLANS' models and their associated input source files contain many of the same data items. Additionally, a variety of relationships exists among the input data. This situation demands that DCSPLANS' use a variety of validation techniques to insure the accuracy of data used by its models. In addition to routine format checks, a series of reasonableness checks are also needed to guarantee that input is both complete and consistent. Reasonableness checks are more complex than the format checks, and are, in

fact, the real key to insuring a truly integrated validation process (i.e., data elements, records and files are not only valid by themselves, but also in relation to other relevant elements, records and files). Of course, validation of the legality and proper sequencing of an input activity itself must precede the validity checks on the data.

An ideal validation tool for DCSPLANS is the active data dictionary. Configured as a data filter, the dictionary provides a flexible, user-friendly, easily expandable validation system for a "small" user group. The data filter can be developed locally using the expertise currently available within DCSPLANS. Such local development allows the data filter system to be tailored precisely to DCSPLANS' own validation needs. The data dictionary approach permits quick, easy adaptation of the data filter to changes in models and input data source files by simply adjusting dictionary metadata. No extensive validation program re-writes will be required. Also, the use of a metadata base as a single source of data for building EVR provides a ready-made mechanism for keeping the EVR consistent. Lastly, an active data dictionary allows DCSPLANS to develop future data processing tools/capabilities with relative ease and minimal investments of time and money.

Preliminary planning is crucial to DCSPLANS' successful development of the data filter. The overall DCSPLANS data processing environment must be understood, and data definition and associated validation requirements must be comprehensively examined and carefully documented. Thorough accomplishment of these first two phases of development will provide a solid base for both preliminary and detailed system design. Preliminary design should be accomplished through a functional decomposition of major system functions. These major functions must be derived from analysis of phase one and two results, and must satisfy the

achievement of the specific goals and key objectives of the DCSPLANS system.

B. RECOMMENDATIONS

An effective DCSPLANS approach to its data validation problem must key on the concepts/designs presented in this thesis. It is recommended that:

1. DCSPLANS pursue an efficient "local" solution which can be tailored to its specific needs, rather than await or attempt to influence the adoption of an organization-wide validation system.
2. the local solution applied by DCSPLANS be an active Data Dictionary "data filter."
3. DCSPLANS begin development with a prototype system that will validate Enlisted Master File (EMF) data for use in two models. This approach tests the system's ability to differentiate between the degrees of validation required by different models using the same source data file, and also takes advantage of the existing UTPACS program (extracts relevant EMF data). The prototype should use an easy-to-program, easy-to-use relational database management system with a simple query language facility (similar to DEASE II).
4. DCSPLANS appoint a small project team to oversee the data filter development. The team must conduct a thorough on-site review of DCSPLANS environmental characteristics and data definition/validation criteria (Chapter Four) prior to revisions of the general design (Chapter Five) and subsequent coding. While detailed design and coding can be conducted off-site (perhaps as a thesis project), the review of

environmental characteristics, data definition, and validation criteria, must be accomplished at DCSPLANS by personnel familiar with DCSPLANS operations. The checklists in chapter four provide comprehensive guidelines for such an examination.

LIST OF REFERENCES

1. Hanson, Owen, Design of Computer Data Files, Computer Science Press, 1982.
2. Senn, James A., Information Systems In Management, Wadsworth Publishing, 1982.
3. ibid.
4. ibid.
5. Krcenke, David M., Database Processing, Scientific Research Associates, 1983.
6. Appleton, Daniel S., "Business Rules: The Missing Link", Datamation, v. 30, number 16, October 1984.
7. Leong-Hong, E.W. and Plagman, B.K., Data Dictionary/Directory Systems, Wiley, 1982.
8. ibid.
9. ibid.
10. ibid.
11. Sprague, Ralph H. and Carlson, Eric D., Building Effective Decision Support Systems, Prentice-Hall, 1982.
12. Leong-Hong, E.W. and Plagman, B.K., Data Dictionary/Directory Systems, Wiley, 1982.
13. Pressman, Roger S., Software Engineering: A Practitioner's Approach, McGraw-Hill, 1982.

INITIAL DISTRIBUTION LIST

	No.	Copies
1. Defense Technical Information Center Cameron Station, Alexandria, Virginia 22314	2	
2. Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2	
3. Department Chairman, Code 54 Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943	1	
4. Dr. Dan Dolk, Code 54DH Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943	3	
5. Major(P) Robert M. DiBona 19 Revere Road Monterey, California 93940	1	
6. Computer Technology Curriculum Office Code 37 Naval Postgraduate School Monterey, California 93943	1	

END

FILMED

7-85

DTIC